**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФИЛИАЛ ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ**

**ВЫСШЕГО ОБРАЗОВАНИЯ**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МЭИ»**

**В Г. СМОЛЕНСКЕ**

Кафедра **«Вычислительной техники»**

Направление 09.03.01 **«Информатика и вычислительная техника»**

**ОТЧЁТ**

**по исполнительской практике по получению первичных профессиональных умений и навыков, в том числе первичных умений и навыков научно-исследовательской деятельности**

студента (студентки) 1 курса группы

(подпись) (фамилия, инициалы)

Место прохождения практики:

(указать место прохождения практики)

Отчёт сдан «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_г.

Руководитель практики от образовательной организации:

(должность) (подпись) (расшифровка подписи)

Защита отчёта состоялась « »\_ 20 г.

Оценка за практику

(неудовлетворительно, удовлетворительно, хорошо, отлично)

Члены комиссии:

(должность) (подпись) (расшифровка подписи)

(должность) (подпись) (расшифровка подписи)

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_г.

Смоленск 20\_\_

Оглавление

[Теоретическое введение 3](#_Toc516052872)

[Анализ задания 7](#_Toc516052873)

[Практическое задание 7](#_Toc516052874)

[Заключение 10](#_Toc516052875)

[Список литературы 10](#_Toc516052876)

# Теоретическое введение

**Эволюция шаблона Model-View-ViewModel**

Популярные шаблоны проектирования упрощали людям жизнь с первых шагов создания пользовательских интерфейсов программ. Например, шаблон модель-представление-презентатор (MVP) была популярна на различных платформах программирования пользовательских интерфейсов. MVP — это разновидность шаблона модель-представление-контроллер, которому уже несколько десятков лет. Если вам никогда не приходилось использовать шаблон MVP, ниже приведено его краткое описание. То, что видно на экране, — это представление; данные, которые там отображены — это модель, а презентатор объединяет их вместе. Представление нуждается в презентаторе для заполнения данными модели, реакции на ввод пользователя, предоставления проверки ввода (в том числе за счет передачи этой функции модели) и других подобных задач. Если нужны дополнительные сведения о шаблоне «модель-представление-презентатор», я советую прочесть статью Жана Поля Буду (Jean-Paul Boodhoo) [«Шаблоны проектирования» за август 2006 года](https://msdn.microsoft.com/magazine/cc188690).

В 2004 Мартин Фаулер (Martin Fowler) опубликовал статью о шаблоне под названием [Модель презентации](http://martinfowler.com/eaaDev/PresentationModel.html) (PM). Шаблон «модель презентации» похож на MVP в том плане. что он отделяет представление от его поведения и состояния. Любопытная часть шаблона PM в том, что создается абстракция представления, которая называется моделью презентации. Представление, таким образом, становится просто результатом обработки модели презентации. Согласно Фаулеру, модель презентации постоянно обновляет свое представление, поэтому они остаются синхронизированными друг с другом. Эта логика синхронизации существует в виде кода в классах модели презентации.

В 2005 году Джон Госсман (John Gossman), сейчас один из архитекторов WPF и Silverlight в корпорации Microsoft, рассказал в своем блоге о [шаблоне модель-представление-модель представления (MVVM)](http://blogs.msdn.com/johngossman/archive/2005/10/08/478683.aspx). MVVM совпадает с моделью презентации Фаулера в том плане. что оба шаблона содержат абстракцию представления, содержащую состояние и поведение представления. Фаулер ввел модель презентации как способ создания независимого от платформы пользовательского интерфейса абстракции представления, а Госсман предложил MVVM как стандартизированный способ использовать основные функции WPF для упрощения создания пользовательских интерфейсов. В этом смысле я считаю MVVM частным вариантом более общего шаблона PM, приспособленным для платформ WPF и Silverlight.

В прекрасной статье Гленна Блока (Glenn Block) «[Prism: шаблоны для создания составных приложений с помощью WPF](https://msdn.microsoft.com/magazine/cc785479)» в выпуске за сентябрь 2008 года описано руководство Microsoft по составным приложениям для WPF. Термин ViewModel (модель представления) не используется. Для описания абстракции представления используется термин «модель презентации». Однако в этой статье я буду называть шаблон аббревиатурой MVVM, а абстракцию представления — моделью представления. В сообществах WPF и Silverlight такая терминология значительно более распространена.

В отличие от презентатора в MVP, модель представления не нуждается в ссылке на представление. Представление привязывается к свойствам модели представления, которая, в свою очередь, представляет данные в объектах модели и других состояниях, нужных для этого представления. Привязки между представлением и моделью представления создавать легко, потому что объект модели представления устанавливается как контекст DataContext представления. Если изменяются значения в модели представления, эти новые значения автоматически переходят в представление через привязку данных. Когда пользователь нажимает кнопку в представлении, для произведения нужного действия выполняется команда в модели представления. Все изменения данных модели всегда производит модель представления, а не представление.

Классы представления не знают о существовании классов модели, а модель представления и модель не знают о представлении. Модель на самом деле вообще не имеет представления о том, что существуют модель представления и представление. Это очень слабо связанная конструкция, и это дает ряд преимуществ, о которых я вскоре расскажу.

MVVM состоит из трех компонентов: модели (Model), модели представления (ViewModel) и представления (View).

**Model**

Модель описывает используемые в приложении данные. Модели могут содержать логику, непосредственно связанную этими данными, например, логику валидации свойств модели. В то же время модель не должна содержать никакой логики, связанной с отображением данных и взаимодействием с визуальными элементами управления.

Нередко модель реализует интерфейсы INotifyPropertyChanged или INotifyCollectionChanged, которые позволяют уведомлять систему об изменениях свойств модели. Благодаря этому облегчается привязка к представлению, хотя опять же прямое взаимодействие между моделью и представлением отсутствует.

**View**

View или представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Применительно к WPF представление - это код в xaml, который определяет интерфейс в виде кнопок, текстовых полей и прочих визуальных элементов.

Хотя окно (класс Window) в WPF может содержать как интерфейс в xaml, так и привязанный к нему код C#, однако в идеале код C# не должен содержать какой-то логики, кроме разве что конструктора, который вызывает метод InitializeComponent и выполняет начальную инициализацию окна. Вся же основная логика приложения выносится в компонент ViewModel.

Однако иногда в файле связанного кода все может находиться некоторая логика, которую трудно реализовать в рамках паттерна MVVM во ViewModel.

Представление не обрабатывает события за редким исключением, а выполняет действия в основном посредством команд.

**ViewModel**

ViewModel или модель представления связывает модель и представление через механизм привязки данных. Если в модели изменяются значения свойств, при реализации моделью интерфейса INotifyPropertyChanged автоматически идет изменение отображаемых данных в представлении, хотя напрямую модель и представление не связаны.

ViewModel также содержит логику по получению данных из модели, которые потом передаются в представление. И также VewModel определяет логику по обновлению данных в модели.

Поскольку элементы представления, то есть визуальные компоненты типа кнопок, не используют события, то представление взаимодействует с ViewModel посредством команд.

Например, пользователь хочет сохранить введенные в текстовое поле данные. Он нажимает на кнопку и тем самым отправляет команду во ViewModel. А ViewModel уже получает переданные данные и в соответствии с ними обновляет модель.

Итогом применения паттерна MVVM является функциональное разделение приложения на три компонента, которые проще разрабатывать и тестировать, а также в дальнейшем модифицировать и поддерживать.

**Преимущества MVVM**

1. **Тестируемость MVVM-приложений**

Приложения, разработанные с использованием MVVM, обладают очень хорошим основанием для проведения модульного тестирования с целью проверки работы отдельных классов и методов.

1. **Меньшее количество кода**

Объем кода, необходимого для управления представлением немного снижается при использовании MVVM, а это означает, что снижается риск допустить ошибки и уменьшается код для написания модульных тестов.

1. **Улучшенное проектирование приложений**

Разработчики и дизайнеры могут самостоятельно работать над разными частями приложения. Как вы видели на примере, представление генерируется в XAML-разметке и использует базовый синтаксис привязок и команд, для взаимодействия с модель-представлением. Вы можете создать модель-представление, которое предоставляет необходимые точки входа для связывания с представлением (например, общедоступные свойства), которые в конечном представлении можно будет легко привязать. Это позволяет дизайнерам работать над внешним видом приложения, а программистам над бизнес-логикой приложения.

1. **Легкость понимания логики представления**

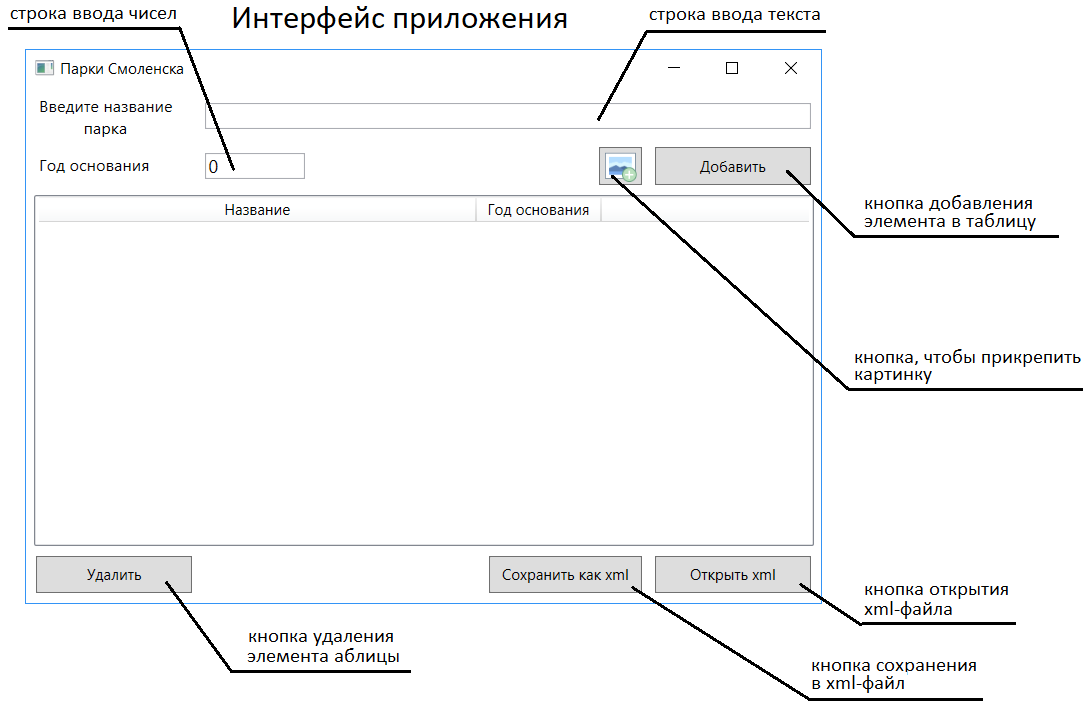
MVVM предусматривает хорошо организованную и легкую для понимания конструкцию построения графического интерфейса за счет использования механизмов привязок, команд и шаблонов данных

# Анализ задания

Суть задания заключается в том, чтобы написать программу на языке C#, используя WPF (Windows Presentation Foundation), основываясь на паттерне MVVM (Model-View-ViewModel). Для того, чтобы решить данное задание следует ознакомится с принципами паттерна MVVM, узнать, чем он полезен при написании программы. Также следует изучить такой язык разметки как XAML.

# Практическое задание

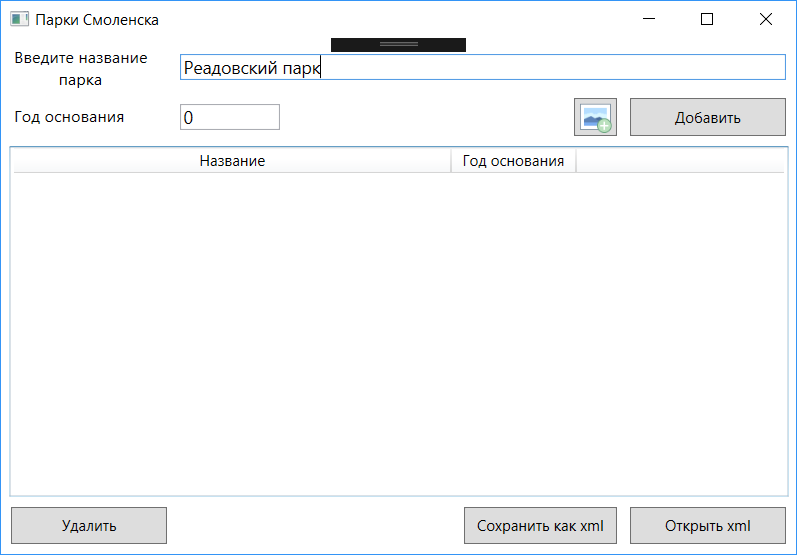
Приложение WPF



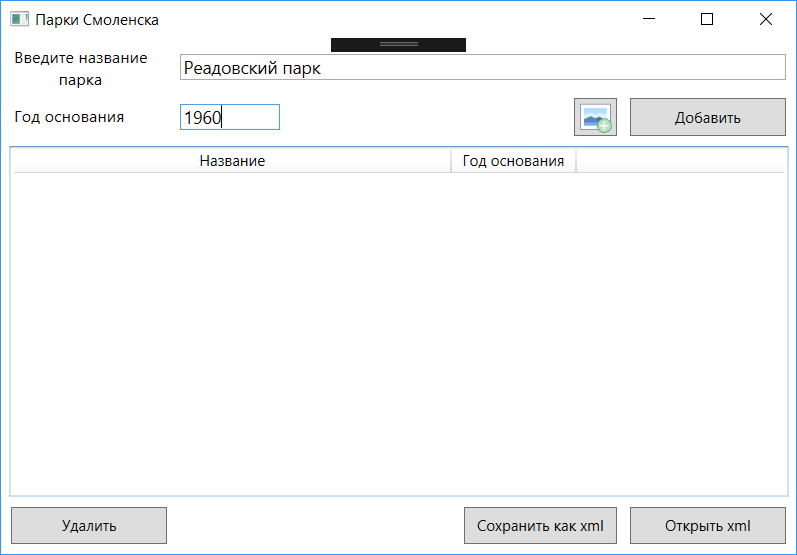
**Инструкция к приложению**

Чтобы добавить парк в таблицу:

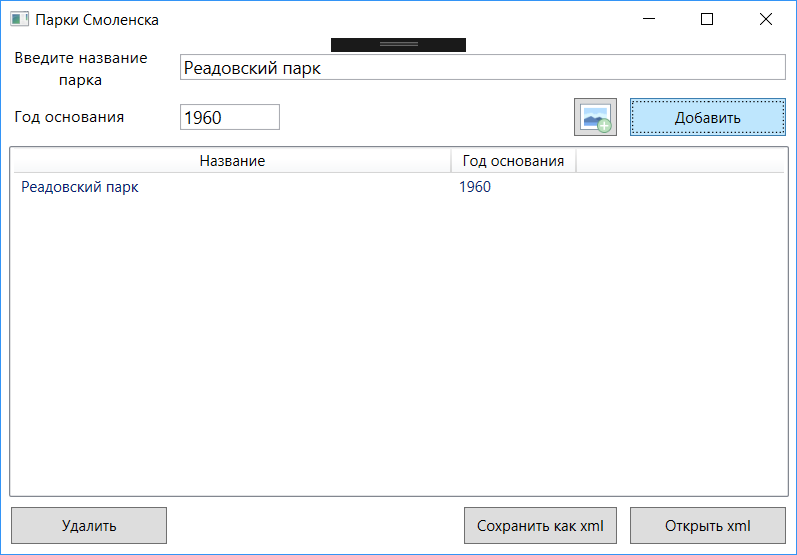
1. Вводим название парка.



1. Вводим год основания парка.

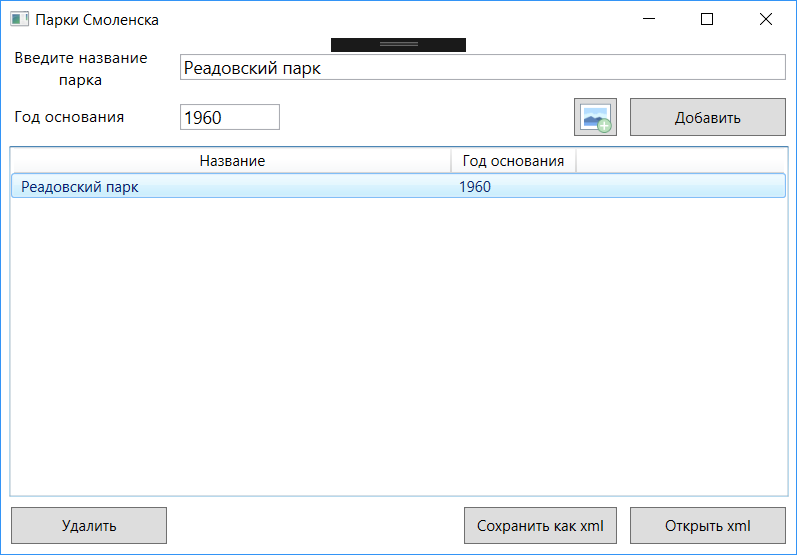


1. Нажимаем кнопку «Добавить».

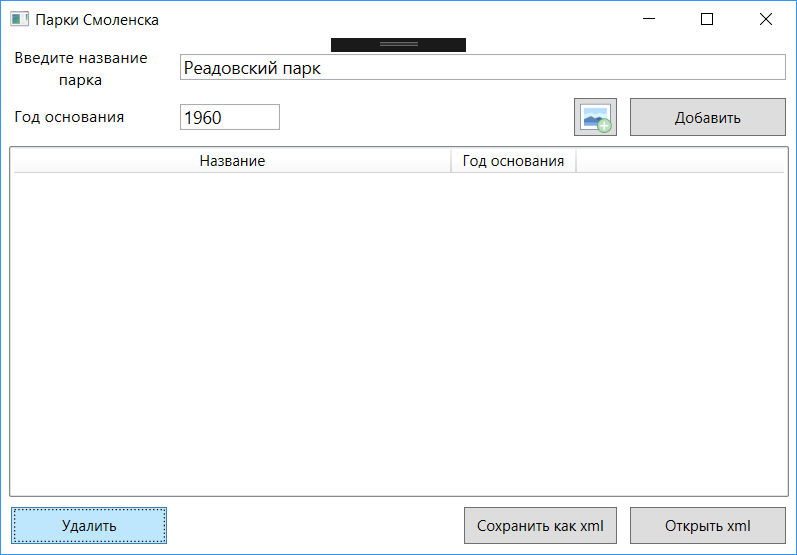


Чтобы удалить парк:

1. Нажимаем левой кнопкой мыши на элемент в таблице.



1. Нажимаем кнопку «Удалить».



# Заключение

В ходе выполнения практики были получены и усовершенствованы навыки работы на таких языках программирования как C++, C#. Были приобретены применены на практике навыки работы с WPF и паттерном MVVM. Было произведено повторение ранее пройденного лекционного материала.

Список литературы

1. Приложения WPF с шаблоном проектирования модель-представление-модель представления. [электронный ресурс]:

https://msdn.microsoft.com/ru-ru/magazine/dd419663.aspx#id0090009

2. Паттерн MVVM. [электронный ресурс]: https://professorweb.ru/my/WPF/documents\_WPF/level36/36\_5.php

3. Введение в WPF. Особенности платформы WPF. [электронный ресурс]:

https://metanit.com/sharp/wpf/1.php